

BIRZEIT UNIVERSITY  
FACULTY OF ENGINEERING AND TECHNOLOGY  
DEPARTMENT OF COMPUTER SCIENCE

COMP438: Encryption Theory  
Final Project:  
RSA Implementation: using JAVA

Prepared by:  
**Aws Ayyash 1190680**

Instructor: **Dr. Mohammad Alkhanafseh**

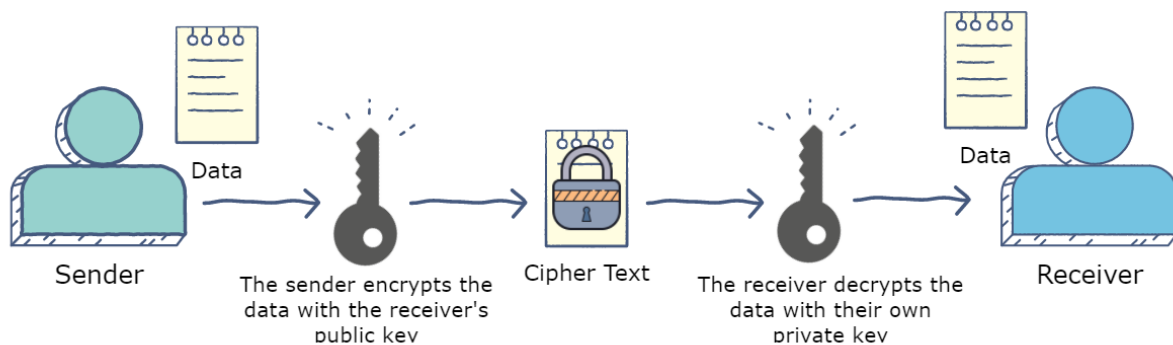
30-jan-2022

<b>Introduction</b>	<b>3</b>
<b>How it works</b>	<b>3</b>
Generating the keys	3
Encryption	4
Decryption	4
<b>My Work</b>	<b>4</b>
Encryption	4
Decryption	7
<b>Actual work (programming)</b>	<b>8</b>
RSA	8
How the keys are generated	8
Encryption	9
The encryption process	9
Decryption	9
The decryption process	9
<b>References</b>	<b>10</b>

# Introduction

The RSA algorithm is an asymmetric cryptography algorithm; this means that it uses a public key and a private key (i.e two different, mathematically linked keys). As their names suggest, a public key is shared publicly, while a private key is secret and must not be shared with anyone. The RSA algorithm is named after those who invented it in 1978: Ron Rivest, Adi Shamir, and Leonard Adleman.

The following illustration highlights how asymmetric cryptography works:



## How it works

### Generating the keys

Select two large prime numbers,  $p$  and  $q$ . The prime numbers need to be large so that they will be difficult for someone to figure out.

Calculate  $n = p * q$ .

Calculate phi:  $\phi(n) = (p-1)(q-1)$ .

Select an integer  $e$ , such that  $e$  is co-prime to  $\phi$ , and  $1 < e < \phi$ . The pair of numbers  $(n, e)$  makes up the public key.

Note: Two integers are co-prime if the only positive integer that divides them is 1.

Calculate  $d$  such that  $e * d \bmod \phi = 1$ .

## Encryption

Given a plaintext  $P$ , represented as a number, the ciphertext  $C$  is calculated as:

$$C = P^e \bmod n.$$

## Decryption

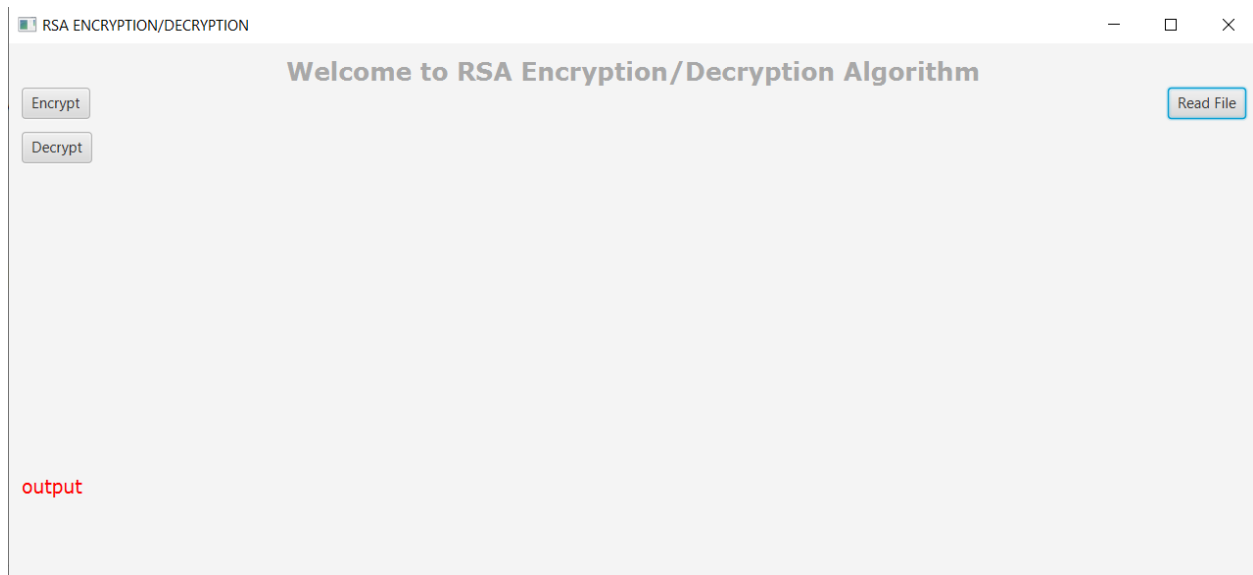
Using the private key  $(n, d)$ , the plaintext  $P$  can be found using:

$$P = C^d \bmod n$$

## My Work

First the implementation is using JAVA and JAVAFX for the graphical user interface.

Below is the application main page:

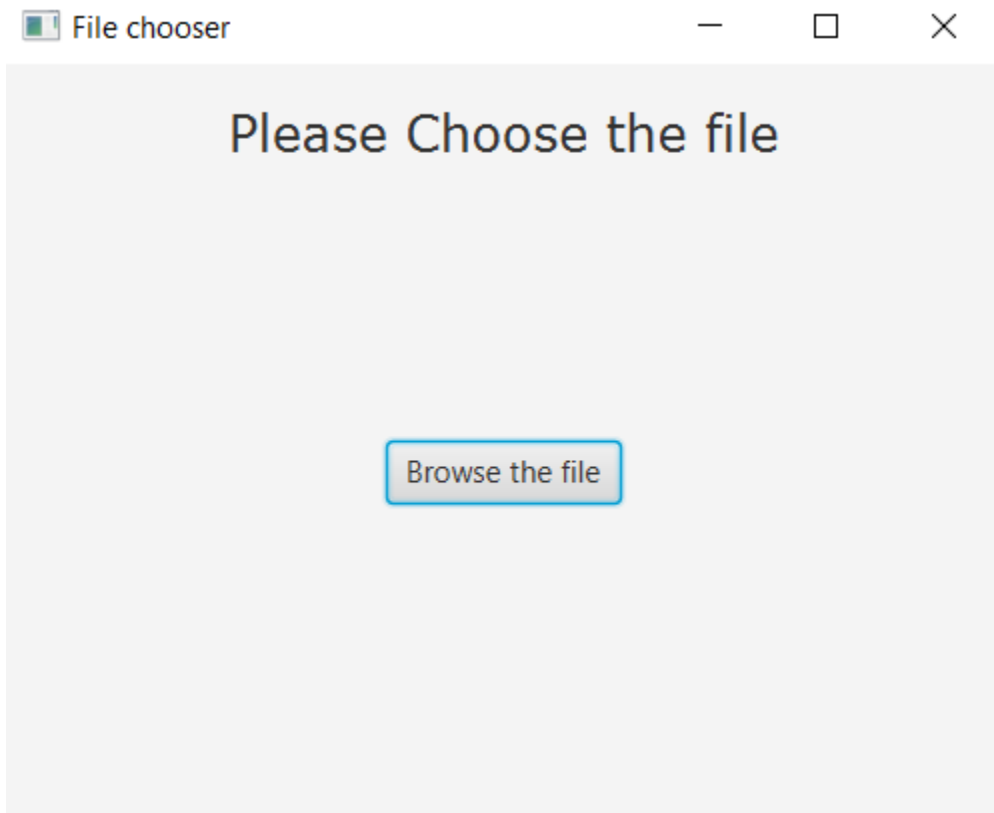


## Encryption


To encrypt a message, the following steps must be followed:

1. Read the file, by clicking the “Read File” option located in the top right corner of the interface.

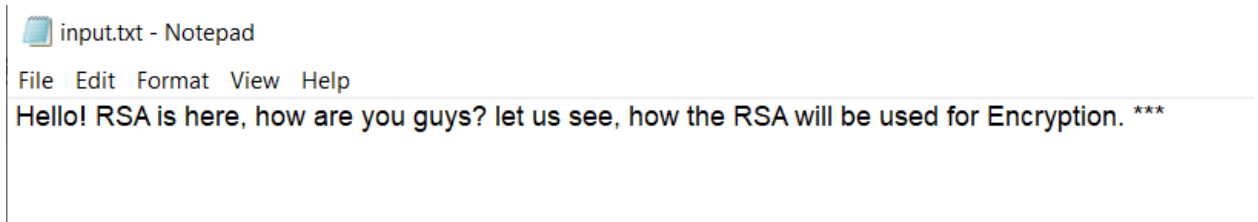
2. A small interface will come up:



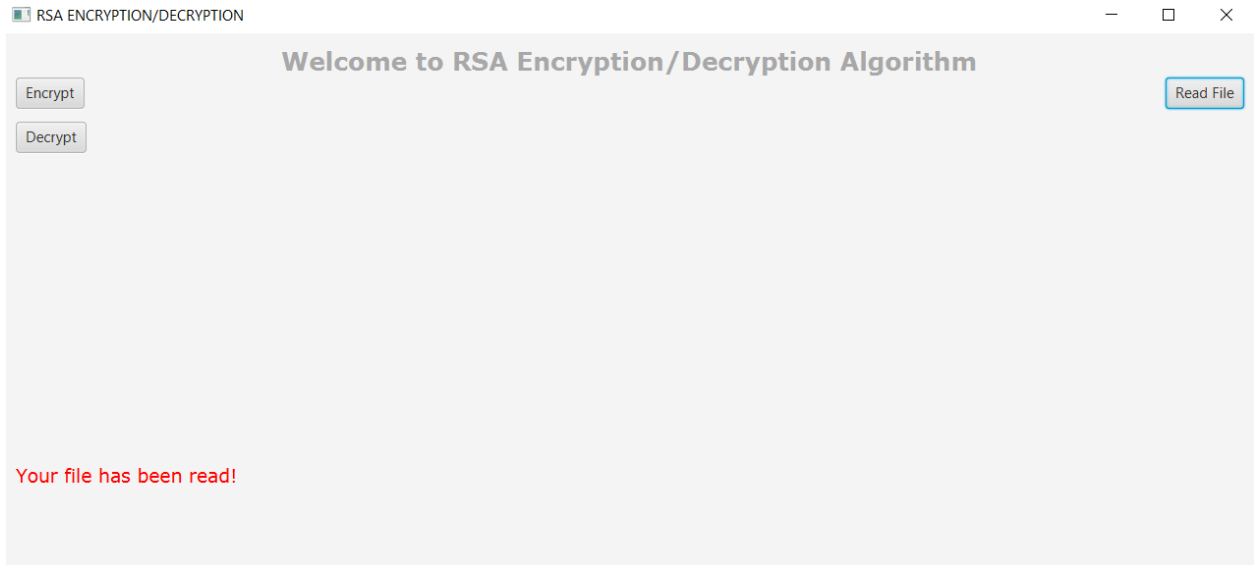
3. Clicking the “Browse the file” option will open your file system explorer:
4. Then choose your input file:

Name	Date modified	Type	Size
 input.txt	1/30/2022 2:16 PM	Text Document	1 KB

and the input file content are:



5. Automatically the small interface will disappear, once the file is selected, and back to the main page:



an output will appear, saying the file is read correctly!

- Now, Click the decrypt option in the top left corner:




The encrypted message (as numbers) will appear on the interface!

- Then we can see that in the same directory (folder) the encrypted message is written to a file called "fileHasEncryptedMsg.txt":

Name	Date modified	Type	Size
fileHasEncryptedMsg.txt	1/30/2022 4:08 PM	Text Document	1 KB
input.txt	1/30/2022 2:16 PM	Text Document	1 KB

And this is its content of the encrypted message:



fileHasEncryptedMsg.txt - Notepad

File Edit Format View Help

115  
437  
265

0 64 419 265 64 265 427 427 312 64 406 21 413 64 171 406 427 64 168 265  
0 64 413 331 410 410 64 1 427 64 419 265 427 432 64 264 21 168 64 427 72 98 168 93 428 171 331 21 72 429 64 213 213 213

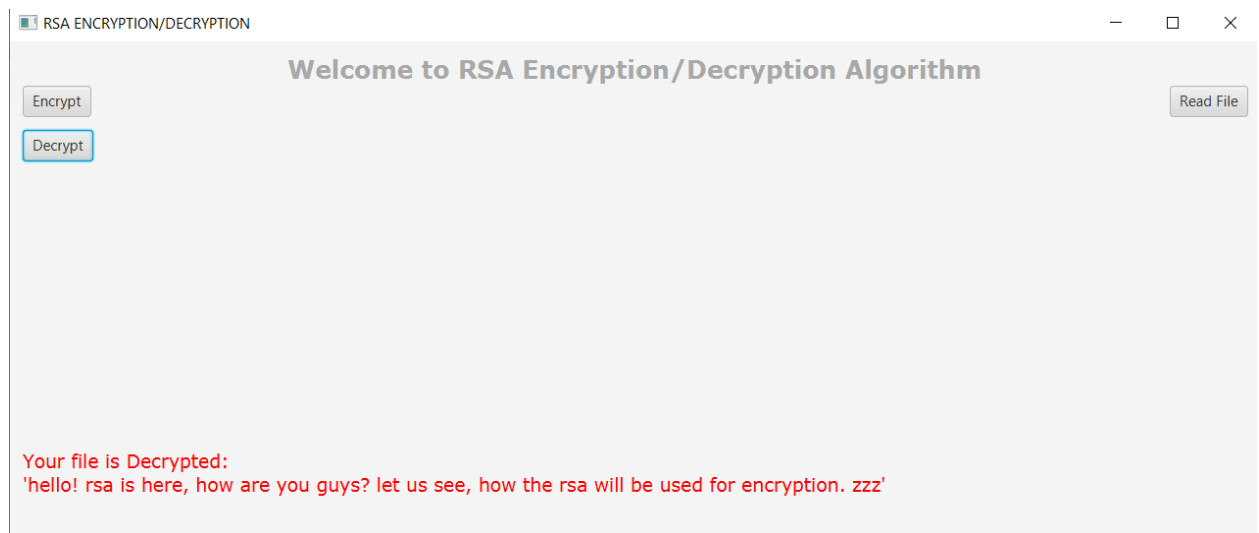
The first and second lines: are the header information (1st: is 'd' the private key) (2nd: is n), to be used later in the decryption process, as this is just a simulation of a real world application (here, I am the sender and the receiver at the same time!) As we mentioned above, the sender encrypts using the public key (variable: e) and the receiver decrypts the message using his/her private key (variable: d), and n: is for the mod operations.

The rest of the lines (they are appearing as many lines, because of the screen size, but they are one line only) are the encrypted message.

## Decryption

To decrypt a message, the following steps must be followed:

1. The same steps from 1 to 5, in the encryption process are the same, regarding the file input (which is now the Encrypted message, not the plain text).
2. Once the main page shows the success message of the file reading, click on the decrypt option from the main page.
3. The main page will show:



The original message (the decryption process result) between single quotes (for clarity only) is the same as the input file, **but** there are many things to mention here, ready? Let's dive in, first it considers only small letters characters (even if the input is capital letters), it only knows the characters: **[a-z]**, special chars: **',' , '?' , ':' , '-' , '\_'** . any other special

char is not considered and replaced by a 'z' character as we see the "\*\*\*" characters are replaced by "zzz".

- Also, this message is written to a file "located in the same directory of the input encrypted file (the encrypted file as input)"

Name	Date modified	Type	Size
fileHasDecryptedMsg.txt	1/30/2022 4:32 PM	Text Document	1 KB
fileHasEncryptedMsg.txt	1/30/2022 4:30 PM	Text Document	1 KB
input.txt	1/30/2022 4:29 PM	Text Document	1 KB

- And its content of the decrypted message is the same as shown on the interface

fileHasDecryptedMsg.txt - Notepad

File Edit Format View Help

hello! rsa is here, how are you guys? let us see, how the rsa will be used for encryption. zzz

## Actual work (programming)

### RSA

#### How the keys are generated

- First, call the function that chooses random prime numbers (p and q), bounded by min and max values, (i choose min=7, max=32 as educational version of the RSA) ((but they are dynamically can be changed only change them (they are constants in the program) and the program automatically works well)).
- Then calculate  $n = p * q$ ; and  $\phi = (p-1) * (q-1)$ ;
- Then find e (the public key used in the encryption process) which is a random prime number  $1 < e < \phi$ ,
- then find d (the private key, used in the decryption process) by using extended euclidean (table method) algorithm to find the GCD between  $\phi$  and e.



## Encryption

- First, getting input file (which contains the plain text message) from the user.
- Read the file line by line, convert each line to lowerCase and for each character  $\Rightarrow$  make an encryption for it, using the RSA (**explained below**),
- Lastly, take  $d$  (private key), and  $n$  (for mod operations) append them in the beginning of the file each on one line.

### The encryption process

- Encryption using the equation  $c = p^e \bmod n$ ,  $c$ : the cipher text,  $p$ : the plain text character,  $e$ : public key,  $n:(p*q)$ ;
- The power and mod operations are done using the modular exponentiation;
- The plain text character is mapped using a constant array which contains: english letters (in lowercase) and the (!, ?, -, . , and the ' ') mapping using its index, (e.g.  $a=0$ ,  $b=1$ , ...,  $z = 25$ , ...,  $?=28$ , ...);

## Decryption

- First, getting the input file (which contains the encrypted message) from the user.
- Read the first line which is  $d$  (the private key) and the second line which is  $n$  (for the mod operations),
- Then all the encrypted message is in one line (which is the 3rd line) each character is separated by a space “-” character  $\Rightarrow$  split the line based on the space  $\Rightarrow$  so for each character  $\Rightarrow$  make a decryption, using the RSA (**explained below**).

### The decryption process

- Decryption using the equation  $p = c^d \bmod n$ ,  $c$ : the cipher text,  $p$ : the plain text character,  $d$ : private key,  $n:(p*q)$ ;
- The power and mod operations are done using the modular exponentiation;
- Lastly, each value returned from the decryption is a number  $\Rightarrow$  so we need to convert (mapping) the number using the letters (mentioned above in the encryption process (the same one)) array.

## References

- Course book and slides
- [Educative](#)